Эффективная оценка представлений слов в векторном пространстве

Краткая выдержка

Мы предлагаем новые архитектуры моделей для вычисления непрерывных векторных представлений слов из очень больших наборов данных. Качество этих представлений измеряется в задаче о сходстве слов, и результаты сравниваются с ранее наиболее эффективными методами, основанными на различных типах нейронных сетей. Мы наблюдаем значительное повышение точности при гораздо меньших вычислительных затратах, т.е. для того, чтобы выучить высококачественный вектор слов на набор данных из 1,6 миллиарда слов, требуется меньше суток. Кроме того, мы показываем, что эти векторы обеспечивают самую современную производительность в нашем тестовом наборе для измерения синтаксического и семантического сходства слов.

1 Вступление

Многие современные системы и техники НЛП рассматривают слова как атомарные единицы - между словами нет понятия сходства, поскольку они представлены в словаре в виде индексов. У этого выбора есть несколько веских причин - простота, надежность и тот факт, что простые модели, обученные на огромных объемах данных, превосходят сложные системы, обученные на меньшем объеме данных. Примером может служить популярная модель N-gram, используемая для статистического моделирования языка - сегодня можно обучать N-граммы практически на всех доступных данных (триллионах слов). Однако во многих задачах применение простых методов ограничено. Например, объем релевантных данных в предметной области для автоматического распознавания речи ограничен - производительность обычно зависит от объема высококачественных расшифрованных речевых данных (часто это всего лишь миллионы слов). При машинном переводе существующие корпуса для многих языков содержат всего несколько миллиардов слов или даже меньше. Таким образом, бывают ситуации, когда простое расширение базовых методов не приведет к какому-либо значительному прогрессу, и мы должны сосредоточиться на более продвинутых методах. С развитием методов машинного обучения в последние годы стало возможным обучать более сложные модели на гораздо большем

наборе данных, и они, как правило, превосходят простые модели. Вероятно, наиболее успешной концепцией является использование распределенных представлений слов. Например, языковые модели, основанные на нейронных сетях, значительно превосходят N-граммовые модели.

1.1 Цели работы

Основная цель этой статьи - представить методы, которые могут быть использованы для изучения высококачественных векторов слов из огромных наборов данных, содержащих миллиарды слов, и с миллионами слов в словарном запасе. Насколько нам известно, ни одна из ранее предложенных архитектур не была успешно обучена более чем на нескольких сотнях миллионов слов при скромной размерности векторов слов между ними.

Мы используем недавно предложенные методы измерения качества результирующих векторных представлений, предполагая, что не только похожие слова будут иметь тенденцию быть близкими друг к другу, но и что слова могут иметь **несколько степеней сходства** (*ориг. multiple degrees of similarity*). Это было замечено ранее в контексте флективных языков - например, существительные могут иметь несколько окончаний, и если мы будем искать похожие слова в подпространстве исходного векторного пространства, то можно найти слова, которые имеют похожие окончания.

Несколько неожиданно было обнаружено, что сходство словесных репрезентаций выходит за рамки простых синтаксических закономерностей. Используя метод смещения слов, при котором над векторами слов выполняются простые алгебраические операции, было показано, например, что вектор (”Король”) - вектор (”Мужчина”) + вектор (”Женщина”) дает вектор, наиболее близкий к векторному представлению слова ”Королева“.

В этой статье мы пытаемся максимально повысить точность этих векторных операций, разрабатывая новые архитектуры моделей, которые сохраняют линейные закономерности между словами. Мы разрабатываем новый комплексный тест набор предназначен для измерения как синтаксических, так и семантических закономерностей, и показывает, что многие такие закономерности могут быть изучены с высокой точностью. Кроме того, мы обсуждаем, как время обучения и точность зависят от размерности векторов слов и объема обучающих данных.

1.2 Предыдущая работа

Представление слов в виде непрерывных векторов имеет долгую историю. Очень популярная модельная архитектура для оценки языковой модели нейронной сети (NNLM) была предложена в [1], где нейронная сеть прямого действия с линейным проекционным слоем и нелинейным скрытым слоем использовалась для совместного изучения векторного представления слов и статистической языковой модели. За этой работой последовали многие другие.

Еще одна интересная архитектура NNLM была представлена в [13, 14], где векторами слов сначала изучается с использованием нейронной сети с одним скрытым слоем. Затем векторы слов используются для обучения NNLM. Таким образом, векторы слов изучаются даже без создания полной NNLM. В этой работе мы непосредственно расширяем эту архитектуру и фокусируемся только на первом шаге, на котором изучаются векторы слов с использованием простой модели.

Позже было показано, что слово "векторы" может быть использовано для значительного улучшения и упрощения многих задач. Приложения НЛП. Оценка самих словесных векторов проводилась с использованием различных архитектур и обучались работе с различными корпусами, и некоторые из полученных векторов слов были доступны для будущих исследований и сравнений. Однако, насколько нам известно, эти архитектуры были значительно более затратными с точки зрения вычислений для обучения, чем предложенная в [13], за исключением определенной версии логарифмически-билинейной модели, где используются диагональные весовые матрицы.

2 Архитектура модели

Для оценки непрерывных представлений слов было предложено множество различных типов моделей, включая хорошо известный латентный семантический анализ (LSA) и латентное распределение Дирихле (LDA). В этой статье мы сосредоточимся на распределенных представлениях слов, изучаемых нейронными сетями, поскольку ранее было показано, что они значительно лучше, чем LSA, сохраняют линейные закономерности между словами [20, 31]; кроме того, LDA становится очень дорогостоящим с точки зрения вычислений при больших наборах данных.

Как и в [18], для сравнения различных архитектур моделей мы сначала определяем вычислительную сложность модели как количество параметров, к которым необходимо получить доступ для полного обучения модели. Далее мы попытаемся максимизировать точность при минимизации вычислительной сложности.

Для всех следующих моделей сложность обучения пропорциональна

$$O = E \times T \times Q, \quad (1)$$

где E - количество периодов обучения, T - количество слов в обучающем наборе, а Q определяется дополнительно для каждой архитектуры модели. Обычно выбирают E = 3 - 50 и T до одного миллиарда. Все модели обучаются с использованием стохастического градиентного спуска и обратного распространения.

2.1 Языковая модель нейронной сети прямого действия (NNLM)

Вероятностная модель языка нейронной сети прямого действия была предложена в [1]. Она состоит из входного, проекционного, скрытого и выходного уровней. На входном уровне N предыдущих слов кодируются с использованием кодирования *1-of-V*, где V - размер словарного запаса. Затем входной слой проецируется на проекционный слой P, который имеет размерность N × D, используя общую проекционную матрицу. Поскольку в любой момент времени активны только N входных данных, компоновка проекционного слоя является относительно дешевой операцией.

Архитектура NNLM становится сложной для вычислений между проекцией и скрытым слоя, поскольку значения в проекционном слое являются плотными. При обычном выборе N = 10 размер проекционного слоя (P) может составлять от 500 до 2000 единиц, в то время как размер скрытого слоя H обычно составляет от 500 до 1000 единиц. Более того, скрытый слой используется для вычисления распределения вероятностей по всем словам всловаре, в результате чего получается выходной слой с размерностью V. Таким образом, вычислительная сложность для каждого обучающего примера составляет

$$Q = N \times D + N \times D \times H + H \times V, \quad (2)$$

где доминирующий член равен H × V. Однако было предложено несколько практических решений, позволяющих избежать этого; либо использовать иерархические версии softmax, либо полностью избегать нормализованных

моделей, используя модели, которые не нормализуются во время обучения. При бинарном древовидном представлении словаря количество выходных единиц, которые необходимо вычислить, может составлять около log2(V). Таким образом, большая часть сложности связана с термином N × D × H.

В наших моделях мы используем иерархический softmax, где словарь представлен в виде двоичного дерева Хаффмана. Это следует из предыдущих наблюдений о том, что частота слов хорошо подходит для получения классов в языковых моделях нейронных сетей. Деревья Хаффмана присваивают короткие двоичные коды часто встречающимся словам, и это еще больше сокращает количество выходных блоков, которые необходимо вычислять: в то время как для сбалансированного двоичного дерева требуется вычислять выходные данные по логарифму log2(V), иерархический softmax на основе дерева Хаффмана требует всего около log2 (*Unigram_perplexity*(V)). Например, если размер словарного запаса равен одному миллиону, это приводит к ускорению оценки примерно в два раза. Хотя это не является существенным ускорением для нейросетевых сетей, поскольку узкое место в вычислениях находится в терминах N × D×H, позже мы предложим архитектуры, которые не имеют скрытых уровней и, следовательно, сильно зависят от эффективности нормализации softmax.

2.2 Рекуррентная языковая модель нейронной сети (RNNLM)

Языковая модель, основанная на рекуррентной нейронной сети, была предложена для преодоления определенных ограничений NNLM с прямой связью, таких как необходимость указывать длину контекста (порядок модели N), а также потому, что теоретически RNN могут эффективно представлять более сложные паттерны, чем неглубокие нейронные сети. Модель RNN не имеет проекционного слоя; только входной, скрытый и выходной слои. Особенностью моделей этого типа является рекуррентная матрица, которая соединяет скрытый слой с самим собой, используя соединения с задержкой по времени. Это позволяет рекуррентной модели формировать своего рода кратковременной памяти, поскольку информация из прошлого может быть представлена состоянием скрытого слоя, которое обновляется на основе текущего ввода и состояния скрытого слоя на предыдущем временном шаге.

Сложность каждого обучающего примера модели RNN составляет

$$Q = H \times H + H \times V, \quad (3)$$

где словесные представления D имеют ту же размерность, что и скрытый слой H. Опять же, значение H × V может быть эффективно сокращено до H × log2(V) с помощью иерархического softmax. Тогда большая часть сложности возникает из-за H × H.

2.3 Параллельное обучение нейронных сетей

Для обучения моделей на огромных массивах данных мы внедрили несколько моделей поверх крупномасштабно распределенной платформы DistBelief, включая NNLM с прямой связью и новые модели, предложенные в этой статье. Фреймворк позволяет нам параллельно запускать несколько реплик одной и той же модели, и каждая реплика синхронизирует свои обновления градиента через централизованный сервер, который хранит все параметры. Для этого параллельного обучения мы используем мини-пакетный асинхронный градиентный спуск с адаптивной процедурой скорости обучения, называемой Adagrad. В рамках этой структуры обычно используется сто или более копий моделей, каждая из которых использует множество процессорных ядер на разных компьютерах в центре обработки данных.

3 Новые логарифмические линейные модели

В этом разделе мы предлагаем две новые архитектуры моделей для изучения распределенных представлений слов, которые пытаются свести к минимуму сложность вычислений. Основное замечание из предыдущего раздела заключалось в том, что большая часть сложности вызвана нелинейным скрытым слоем в модели. Хотя именно это делает нейронные сети такими привлекательными, мы решили изучить более простые модели, которые, возможно, не смогут представлять данные так же точно, как нейронные сети, но способные более эффективно обучаться на гораздо большем количестве данных.

Новые архитектуры непосредственно соответствуют тем, которые были предложены в наших предыдущих работах, где было обнаружено, что языковая модель нейронной сети может быть успешно обучена в два этапа: сначала с помощью простой модели изучаются непрерывные векторы слов, а затем N-граммовый NNLM обучается поверх этих распределенных представлений из слов. Несмотря на то, что позднее был проведен значительный объем работы, направленной на изучение векторов слов, мы считаем подход, предложенный в [13], наиболее простым. Обратите внимание, что соответствующие модели были предложены также гораздо раньше.

3.1 Модель непрерывного Bag-of-Words

Первая предложенная архитектура аналогична NNLM с прямой связью, где нелинейный скрытый слой удален, а проекционный слой является общим для всех слов (а не только для матрицы проекции); таким образом, все слова проецируются в одно и то же положение (их векторы усредняются). Мы называем эту архитектуру bag-of-words, поскольку порядок слов в истории не влияет на проекцию. Кроме того, мы также используем слова из будущего; мы добились наилучшего результата в решении этой задачи, представленной в следующем разделе, путем построения логарифмического линейного классификатора с четырьмя будущими и четырьмя историческими словами на входе, где критерием обучения является правильная классификация текущего (среднего) слова.

Сложность обучения тогда

$$Q = N \times D + D \times log2(V). \quad (4)$$

Далее мы обозначаем эту модель как CBOW, поскольку, в отличие от стандартной модели bag-of-words, она использует непрерывное распределенное представление контекста. Архитектура модели показана на рисунке 1. Обратите внимание, что весовая матрица между входным и проекционным слоями является общей для всех позиций слов таким же образом, как и в NNLM.

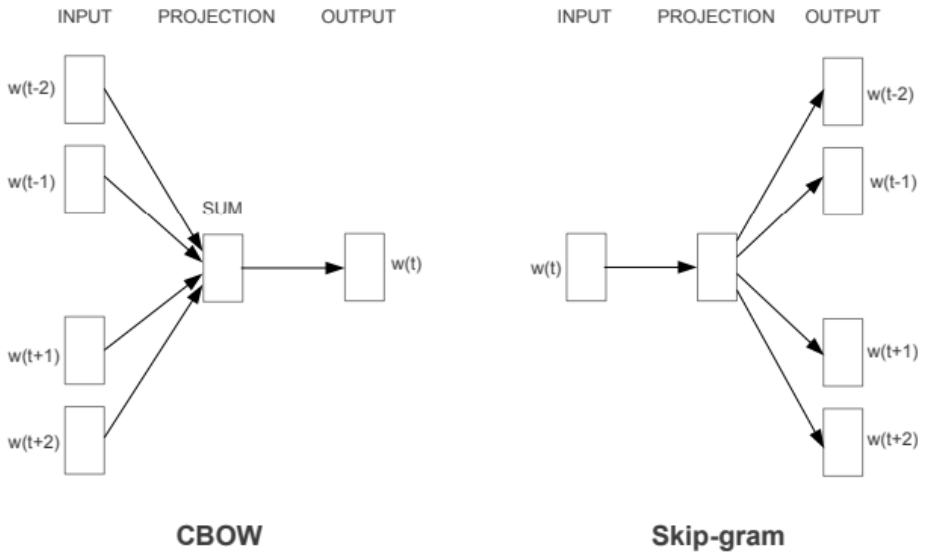

Рисунок 1: Новые архитектуры моделей. Архитектура CBOW предсказывает текущее слово на основе контекста, а Skip-gram предсказывает окружающие слова, заданные текущим словом.

3.2 Модель непрерывного Skip-gram

Вторая архитектура похожа на CBOW, но вместо того, чтобы предсказывать текущее слово на основе контекста, она пытается максимально точно классифицировать слово на основе другого слова в том же предложении. Точнее, мы используем каждое текущее слово в качестве входных данных для логарифмического линейного классификатора с непрерывным проекционным слоем и предсказываем слова в определенном диапазоне до и после текущего слова. Мы обнаружили, что увеличение диапазона улучшает качество результирующих векторов слов, но также увеличивает сложность вычислений. Поскольку более удаленные слова обычно менее связаны с текущим значением, в отличие от близких к нему слов, мы придаем меньшее значение отдаленным словам, делая меньшую выборку из этих слов в наших обучающих примерах.

Сложность обучения в этой архитектуре пропорциональна

$$Q = C \times (D + D \times log2(V)), \quad (5)$$

где C - максимальное расстояние между словами. Таким образом, если мы выберем C = 5, то для каждого обучающего слова мы будем случайным образом выбирать число R в диапазоне <1; C>, а затем использовать слова из истории и R слов из будущего текущего слова в качестве правильных меток. Для этого нам потребуется выполнить классификацию R × 2 слов, используя текущее слово в качестве входных данных и каждое из R + R слов в качестве выходных данных. В следующих экспериментах мы используем C = 10.

4 Результат

Чтобы сравнить качество различных версий **word vectors**, в предыдущих статьях обычно использовалась таблица с примерами слов и наиболее похожими на них словосочетаниями, которые можно понять интуитивно. Хотя легко показать, что слово *France* похоже на *Italy* и, возможно, на некоторые другие страны, гораздо сложнее использовать эти векторы в более сложной задаче на сходство, как показано ниже. Мы следуем предыдущему замечанию о том, что между словами может быть много различных типов сходства, например слово *big* похоже на *bigger* в том же смысле, в каком *small* похоже на *smaller*. Пример другим типом отношений могут быть пары слов " *big - biggest*" и " *small - smallest* " [20]. Далее мы обозначаем две пары слов с одинаковыми отношениями в виде вопроса, поскольку мы можем спросить:” Какое слово похоже на *small* в том же смысле, в каком *biggest* похоже на *big*?"

Несколько удивительно, но на эти вопросы можно ответить, выполнив простые алгебраические операции с векторным представлением слов. Чтобы найти слово, похожее на *small* в том же смысле, в каком *biggest* похоже на *big*, мы можем просто вычислить вектор X = вектор (”*biggest* ”)−вектор (”*big*”) +вектор (”*small* ”). Затем мы ищем в векторном пространстве слово, наиболее близкое к X, измеренное по косинусному расстоянию, и используем его в качестве ответа на вопрос (во время этого поиска мы отбрасываем входные вопросительные слова). Когда векторы слов хорошо обучены, можно найти правильный ответ (слово *smallest*) с использованием этого метода.

Наконец, мы обнаружили, что когда мы тренируем многомерные векторы слов на большом объеме данных, полученные векторы можно использовать для определения очень тонких семантических связей между словами, таких как город и страна, к которой он принадлежит, например, Франция относится к Парижу так же, как Германия - к Берлину. Словесные векторы с такими семантическими связями могут быть использованы для улучшения многих существующих приложений NLP, таких как машинный перевод, системы поиска информации и ответов на вопросы, и могут стать основой для других будущих приложений, которые еще предстоит изобрести.

Таблица 1: *Примеры пяти типов семантических и девяти типов синтаксических вопросов в наборе тестов на семантико-синтаксические отношения слов.*

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

4.1 Описание задачи

Чтобы измерить качество словарных векторов, мы разработали комплексный набор тестов, который содержит пять типов семантических вопросов и девять типов синтаксических вопросов. Два примера из каждой категории приведены в таблице 1. В целом, существует 8869 семантических и 10675 синтаксических вопросов. Вопросы в каждой категории были созданы в два этапа: сначала вручную был создан список похожих пар слов. Затем путем соединения двух пар слов формируется большой список вопросов. Например, мы составили список из 68 крупных американских городов и штатов, к которым они принадлежат, и сформировали около 2,5 тыс. вопросов, выбрав случайным образом две пары слов. Мы включили в наш тестовый набор только односложные слова, поэтому словосочетания, состоящие из нескольких слов, отсутствуют (например, Нью-Йорк).

Мы оцениваем общую точность для всех типов вопросов и для каждого типа вопроса в отдельности (семантический, синтаксический). Предполагается, что на вопрос дан правильный ответ, только если слово, наиболее близкое к вектору, вычисленному с использованием вышеуказанного метода, в точности совпадает с правильным словом в вопросе; синонимы, таким образом, засчитываются как ошибки. Это также означает, что достижение 100%-ной точности, вероятно, невозможно, поскольку текущие модели не содержат никакой входной информации о морфологии слов. Однако мы считаем, что полезность векторов слов для определенных приложений должна положительно коррелировать с этим показателем точности. Дальнейшего прогресса можно достичь, включив информацию о структуре слов, особенно в синтаксические вопросы.

4.2 Максимальное повышение точности

Мы использовали Google News Corpus для обучения вектора слов. Этот корпус содержит около 6B токенов. Мы ограничили объем словаря до 1 миллиона наиболее часто встречающихся слов. Очевидно, что мы сталкиваемся с проблемой оптимизации с ограниченным временем, поскольку можно ожидать, что использование большего количества данных и векторов слов с большей размерностью повысит точность. Чтобы оценить наилучший выбор архитектуры модели для быстрого получения максимально хороших результатов, мы сначала оценили модели, обученные на подмножествах обучающих данных, со словарным запасом, ограниченным наиболее часто встречающимися 30 тысячами слов. Результаты использования архитектуры

CBOW с различным выбором размерности вектора слов и увеличением объема обучающих данных приведены в таблице 2.

Можно видеть, что через некоторое время добавление большего количества измерений или обучающих данных приводит к уменьшению результативности. Итак, мы должны увеличить как размерность вектора, так и объем обучающих данных вместе взятых. Хотя это наблюдение может показаться тривиальным, следует отметить, что в настоящее время популярно обучать векторы слов на относительно больших объемах данных, но с недостаточным размером (например, 50–100). Учитывая уравнение 4, увеличение объема обучающих данных в два раза приводит примерно к такому же увеличению вычислительной сложности, как и увеличение размера вектора в два раза.

Для экспериментов, представленных в таблицах 2 и 4, мы использовали три периода обучения со стохастическим градиентным спуском и обратным распространением. Мы выбрали начальную скорость обучения 0,025 и линейно снижали ее, чтобы в конце последнего периода обучения она приблизилась к нулю.

Таблица 2: *Точность в подмножестве тестов на семантико-синтаксическую взаимосвязь слов с использованием векторов слов из архитектуры CBOW с ограниченным словарным запасом. Используются только вопросы, содержащие слова из 30 тыс. наиболее часто встречающихся слов.*

Dimensionality / Training words	24M	49M	98M	196M	391M	783M
50	13.4	15.7	18.6	19.1	22.5	23.2
100	19.4	23.1	27.8	28.7	33.4	32.2
300	23.2	29.2	35.3	38.6	43.7	45.9
600	24.0	30.1	36.5	40.8	46.6	50.4

Таблица 3: Сравнение архитектур, использующих модели, обученные на одних и тех же данных, с 640-мерными векторами слов. Точность приведена в нашем наборе тестов на семантико-синтаксическую взаимосвязь слов и в наборе тестов на синтаксическую взаимосвязь.

Model Architecture	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness Test Set [20]
	Semantic Accuracy [%]	Syntactic Accuracy [%]	
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

4.3 Сравнение архитектур моделей

Сначала мы сравниваем различные архитектуры моделей для получения векторов слов, используя одни и те же обучающие данные и используя одинаковую размерность в 640 векторов слов. В дальнейших экспериментах мы используем полный набор вопросов из нового набора тестов на семантико-синтаксические отношения слов, т.е. не ограниченный словарным запасом в 30 тысяч слов. Мы также включаем результаты теста, представленного в [20], который фокусируется на синтаксическом сходстве между словами.

Обучающие данные состоят из нескольких корпусов LDC и подробно описаны в [18] (320 миллионов слов, 82 тысячи словарного запаса). Мы использовали эти данные для сравнения с ранее обученной языковой моделью рекуррентной нейронной сети, обучение которой на одном процессоре заняло около 8 недель. Мы обучили NNLM с прямой связью с тем же количеством 640 скрытых единиц, используя параллельное обучение Disbelief, используя историю из 8 предыдущих слов (таким образом, NNLM имеет больше параметров, чем RNNLM, поскольку проекционный слой имеет размер 640 × 8).

В таблице 3 видно, что векторы слов из RNN (используемые в [20]) в основном хорошо отвечают на синтаксические вопросы. Векторы NNLM работают значительно лучше, чем RNN – это неудивительно, поскольку векторы слов в RNNLM напрямую связаны с нелинейным скрытым слоем. Архитектура CBOW лучше, чем NNLM, справляется с синтаксическими задачами и примерно так же с семантическими. Наконец, архитектура Skip-gram немного хуже справляется с синтаксической задачей, чем модель CBOW (но все же лучше, чем NNLM), и намного лучше в семантической части теста, чем все остальные модели.

Затем мы оценили наши модели, обученные с использованием только одного процессора, и сравнили результаты с общедоступными векторами слов. Сравнение приведено в таблице 4. Модель CBOW была обучена на основе подмножества данных Google News примерно за день, в то время как время обучения модели Skip-gram составило около трех дней.

Для экспериментов, о которых мы расскажем далее, мы использовали только один период обучения (опять же, мы линейно снижаем скорость обучения, чтобы в конце обучения она приближалась к нулю). Обучение модели на вдвое большем объеме данных с использованием одной эпохи дает сопоставимые или лучшие результаты, чем повторение одних и тех же данных в течение трех эпох, как показано в таблице 5, и обеспечивает дополнительное небольшое ускорение.

Таблица 4: *Сравнение общедоступных словарных векторов из набора тестов на семантико-синтаксическую взаимосвязь слов и словарных векторов из наших моделей. Используются полные словари.*

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
Our NNLM	20	6B	12.9	26.4	20.3
Our NNLM	50	6B	27.9	55.8	43.2
Our NNLM	100	6B	34.2	**64.5**	50.8
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	**50.0**	55.9	**53.3**

Таблица 5: *Сравнение моделей, подготовленных для трех эпох на основе одних и тех же данных, и моделей, подготовленных для одной эпохи. Точность приведена для полного набора семантико-синтаксических данных.*

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days]
			Semantic	Syntactic	Total	
3 epoch CBOW	300	783M	15.5	53.1	36.1	1
3 epoch Skip-gram	300	783M	50.0	55.9	53.3	3
1 epoch CBOW	300	783M	13.8	49.9	33.6	0.3
1 epoch CBOW	300	1.6B	16.1	52.6	36.1	0.6
1 epoch CBOW	600	783M	15.4	53.3	36.2	0.7
1 epoch Skip-gram	300	783M	45.6	52.2	49.2	1
1 epoch Skip-gram	300	1.6B	52.2	55.1	53.8	2
1 epoch Skip-gram	600	783M	56.7	54.5	55.5	2.5

4.4 Крупномасштабное параллельное обучение моделей

Как упоминалось ранее, мы реализовали различные модели в распределенном фреймворке под названием DistBelief. Ниже мы приводим результаты нескольких моделей, обученных на наборе данных Google News 6B, с использованием асинхронного градиентного спуска в мини-пакете и адаптивной процедуры повышения скорости обучения под названием Adagrad. Во время обучения мы использовали от 50 до 100 копий моделей. Количество процессорных ядер является приблизительным, поскольку компьютеры центра обработки данных используются совместно с другими производственными задачами, и их использование может сильно варьироваться. Обратите внимание, что из-за накладных расходов распределенной платформы загрузка процессора модель CBOW и модель Skip-gram гораздо ближе друг к другу, чем их реализации на простых машинных моделях. Результаты представлены в таблице 6.

Таблица 6: *Сравнение моделей, обученных с использованием распределенной платформы Disbelief. Обратите внимание, что обучение NNLM с использованием 1000-мерных векторов заняло бы слишком много времени.*

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

Таблица 7: *Сравнение и комбинирование моделей на Microsoft Sentence Completion Challenge.*

Architecture	Accuracy [%]
4-gram [32]	39
Average LSA similarity [32]	49
Log-bilinear model [24]	54.8
RNNLMs [19]	55.4
Skip-gram	48.0
Skip-gram + RNNLMs	**58.9**

4.5 Microsoft Research Sentence Completion Challenge

Задача Microsoft Sentence Completion Challenge была недавно введена в качестве задачи для продвижения языкового моделирования и других методов НЛП. Это задание состоит из 1040 предложений, в каждом из которых отсутствует одно слово, и цель состоит в том, чтобы выбрать слово, которое наиболее согласуется с остальной частью предложения, из списка из пяти разумных вариантов. В этом наборе уже сообщалось об эффективности нескольких методов, включая N-граммовые модели, модель на основе LSA, логарифмически-билинейную модель и комбинацию рекуррентных нейронных сетей, которая в настоящее время соответствует современному уровню техники, точность в этом тесте составляет 55,4%.

Мы изучили эффективность архитектуры Skip-gram в решении этой задачи. Сначала мы обучаем 640-мерную модель на основе 50 миллионов слов, приведенных в [32]. Затем мы вычисляем оценку каждого предложения в тестовом наборе, используя неизвестное слово на входе, и прогнозируем все окружающие слова в предложении. Итоговая оценка предложения представляет собой сумму этих отдельных прогнозов. Используя оценки предложений, мы выбираем наиболее вероятное предложение.

Краткое изложение некоторых предыдущих результатов вместе с новыми результатами представлено в таблице 7.

Хотя модель Skip-gram сама по себе не справляется с этой задачей лучше, чем модель подобия LSA, оценки, полученные с помощью этой модели, дополняют оценки, полученные с помощью RNNLMs, а взвешенная комбинация приводит к новому результату с точностью 58,9% (59,2% в части разработки набора и 58,7% от тестовой части набора).

5 Примеров усвоенных отношений

В таблице 8 приведены слова, которые находятся в различных отношениях. Мы следуем описанному выше подходу: связь определяется путем вычитания двух векторов слов, а результат добавляется к другому слову. Таким образом, например, Париж - Франция + Италия = Рим. Как видно, точность довольно хорошая, хотя при условии точного совпадения результаты, приведенные в таблице 8, составили бы всего около 60%). Мы считаем, что векторы слов, обработанные на еще больших наборах данных с большей размерностью, будут работать значительно лучше и позволят разрабатывать новые инновационные приложения. Еще одним способом повышения точности

является приведем более одного примера взаимосвязи. Используя десять примеров вместо одного для формирования вектора взаимосвязи (мы усредняем отдельные векторы вместе), мы наблюдали повышение точности наших лучших моделей примерно на 10% в семантико-синтаксическом тесте.

Также можно применять векторные операции для решения различных задач. Например, мы наблюдали хорошую точность выбора слов, не входящих в список, путем вычисления среднего вектора для списка слов и нахождения наиболее удаленного вектора слов. Это популярный тип задач в некоторых тестах на интеллект человека. Очевидно, что с помощью этих методов еще предстоит сделать много открытий.

Таблица 8: *Примеры парных соотношений слов с использованием наилучших векторов слов из таблицы 4 (модель Skip gram, обученная на 783 миллионах слов с 300 размерностями).*

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

6 Заключение

В этой статье мы изучали качество векторных представлений слов, полученных с помощью различных моделей, на основе набора синтаксических и семантических языковых заданий. Мы обнаружили, что можно обучать высокому качественные векторы слов, использующие очень простую архитектуру моделей по сравнению с популярными моделями нейронных сетей (как прямолинейными, так и рекуррентными). Благодаря значительно меньшей вычислительной сложности, можно вычислять очень точные многомерные векторы слов из гораздо большего набора данных. Используя распределенный фреймворк Disbelief, стало возможно обучать модели CBOW и Skip-gram даже на корпусах с одним триллионом слов, что обеспечивает практически неограниченный объем словарного запаса. Это на несколько

порядков больше, чем лучшие ранее опубликованные результаты для аналогичных моделей.

Интересной задачей, в которой, как недавно было показано, векторы слов значительно превосходят предыдущий уровень техники, является задача 2 полугодия 2012 года. Общедоступные векторы RNN были использованы вместе с другими методами для достижения более чем 50%-ного увеличения ранговой корреляции Спирмена по сравнению с предыдущим лучшим результатом. Векторы слов, основанные на нейронных сетях, ранее применялись для решения многих других задач NLP, например, для анализа настроений и определения перефразирования. Можно ожидать, что эти приложения могут извлечь выгоду из архитектуры моделей, описанных в этой статье.

Наша текущая работа показывает, что векторы слов могут успешно применяться для автоматического расширения фактов в базах знаний, а также для проверки правильности существующих фактов. Результаты экспериментов по машинному переводу также выглядят очень многообещающими. В будущем было бы также интересно сравнить наши методы с латентным реляционным анализом и другими. Мы считаем, что наш комплексный набор тестов поможет исследовательскому сообществу усовершенствовать существующие методы оценки векторов слов. Мы также ожидаем, что высококачественные векторы слов станут важным строительный блок для будущих приложений НЛП.

7 Последующая работа

После того, как была написана первоначальная версия этой статьи, мы опубликовали одномашинный многопоточный код на C++ для вычисления векторов слов, используя как непрерывный пакет слов, так и архитектуру skip-gram. Скорость обучения значительно выше, чем сообщалось ранее в этой статье, т.е. она составляет порядка миллиардов слов в час для типичных вариантов гиперпараметров. Мы также опубликовали более 1,4 миллиона векторов, представляющих именованные объекты, обучены более чем 100 миллиардам слов. Некоторые из наших последующих работ будут опубликованы в предстоящей статье NIPS 2013.